Application and system benchmarks

High-Performance Computing System Administration

Johannes Richter
24.02.2023

Overview

1. Introduction
2. Unified European Applications Benchmark Suite
3. Exemplar Benchmark
    a. Description
    b. Datasets
    c. Dependencies and Installation
    d. Example output
4. References

Introduction

What are Benchmarks(Computing)?

- applications designed to characterize the performance of a system
- results can be used to compare and rank different systems
- to identify problems and monitor the progress of fixing them
  - check if the performance requested by the customer is actually achieved
- usually cannot fully characterize all performance aspects of a system
  - measure one specific metric (floating point operations, execution time,I/O operations...)
- to cover multiple metrics we have benchmark-suites
- many types of benchmarks (Algorithmic benchmarks, Parallel benchmarks,...)

Introduction

Benchmark Key Properties

1. **Relevance:** should measure important features.
2. **Representativeness:** should be broadly accepted by industry and academia.
3. **Equity:** all systems should be fairly compared.
4. **Repeatability:** results should be verifiable.
5. **Cost-effectiveness:** tests should be economical.
6. **Scalability:** tests should measure from single server to multiple servers
7. **Transparency:** Benchmark metrics should be readily understandable.

UEABS-Unified European Applications Benchmark Suite

- a set of currently 13 application codes taken from the pre-existing
  - PRACE and DEISA application benchmark suites
  - extended with the PRACE Accelerator Benchmark Suite
- providing a single benchmark suite of
  - currently relevant and publicly available application codes and datasets
  - a size which can realistically be run on large systems
- each application code has either one, or two input datasets
  - Test Case A is designed to run on Tier-1 sized systems
  - Test Case B is designed to run on Tier-0 sized systems
- current release is Version 2.2 (December 20, 2022)
- https://repository.prace-ri.eu/git/UEABS/ueabs

Example Benchmark

GPAW Description

A Projected Augmented Wave code

- electronic structure calculations based on
  - the density functional theory (DFT)
  - the time-dependent density functional theory (TD-DFT)
- DTF allows studies of ground state properties such as energetics and equilibrium geometries
- TD-DFT can be used for calculating excited state properties such as optical spectra
- written in Python and C and parallelized with MPI
- There is also a CUDA-based implementation for GPU systems

Example Benchmark

GPAW Dependencies and Installation

- C compiler with MPI support
- BLAS, LAPACK, BLACS and ScaLAPACK
- for GPAW 20.10.0 Python 3.6-3.9
- NumPy >=1.9
- SciPy >=0.14
- FFTW (Fastest Fourier Transform in the West), Libxc
- ASE (Atomic Simulation Environment)
- Installation choices
  - spack (py-gpaw)
  - manual with autoconf, Libtool and make

Example Benchmark

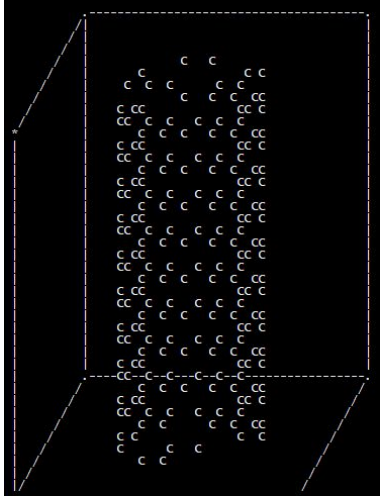
GPAW Datasets

1. Case A (small): Carbon nanotube
   a. ground state calculation for a carbon nanotube in vacuum
   b. scale up to 10 nodes and/or 100 MPI tasks
2. Case B (medium): Copper filament
   a. ground state calculation for a copper filament in vacuum
   b. scale up to 100 nodes and/or 1000 MPI tasks
3. Case C (large): Silicon cluster
   a. ground state calculation for a silicon cluster in vacuum
   b. scale up to 1000 nodes and/or 10000 MPI tasks

- result verification via 4 parameters including
  - expected number of Number of iterations
  - extrapolated energy

```
  ___ ___ ___ _ _ _
 |   |   |_  | | | |
 | | | | | . | | | |
 |__ |  _|___|_____|  21.1.0
 |___|_|

User:   hpctraining13@amp050
Date:   Thu Feb 16 16:27:23 2023
Arch:   x86_64
Pid:    402865
Python: 3.9.0
gpaw:   /usr/users/hpctraining13/.spack/0.17.1/install/haswell/gcc-9.3.0/py-gpaw-21.1.0-div4ac/lib/python3.9/site-packages/gpaw
_gpaw:  /usr/users/hpctraining13/.spack/0.17.1/install/haswell/gcc-9.3.0/py-gpaw-21.1.0-div4ac/lib/python3.9/site-packages/
        _gpaw.cpython-39-x86_64-linux-gnu.so
ase:    /usr/users/hpctraining13/.spack/0.17.1/install/haswell/gcc-9.3.0/py-ase-3.21.1-mqgcim/lib/python3.9/site-packages/ase (version 3.21.1)
numpy:  /usr/users/hpctraining13/.spack/0.17.1/install/haswell/gcc-9.3.0/py-numpy-1.21.3-amwali/lib/python3.9/site-packages/numpy (version 1.21.3)
scipy:  /usr/users/hpctraining13/.spack/0.17.1/install/haswell/gcc-9.3.0/py-scipy-1.7.1-vbxs7k/lib/python3.9/site-packages/scipy (version 1.7.1)
libxc:  4.3.4
units:  Angstrom and eV
cores: 100
OpenMP: False
OMP_NUM_THREADS: 1

Input parameters:
  convergence: {density: 0.01,
                eigenstates: 0.0001,
                energy: 0.001}
  eigensolver: rmm-diis
  h: 0.2
  maxiter: 16
  mixer: {backend: pulay,
          beta: 0.1,
          method: separate,
          nmaxold: 5,
          weight: 50}
  nbands: -60
  occupations: {fixmagmom: False,
                name: fermi-dirac,
                width: 0.1}
  poissonsolver: {eps: 1e-12,
                  name: fast,
                  nn: 3}

System changes: positions, numbers, cell, pbc, initial_charges, initial_magmoms

Initialize ...

C-setup:
  name: Carbon
  id: d60576a1f549371a163e72552ca58787
  Z: 6.0
  valence: 4
  core: 2
  charge: 0.0
  file: /home/uni08/hpctraining13/benchmark_project/ueabs/gpaw/gpaw_data/gpaw-setups-0.9.20000/C.LDA.gz
  compensation charges: gauss, rc=0.20, lmax=2
  cutoffs: 1.14(filt), 1.14(core),
  valence states:
                  energy  radius
    2s(2.00)   -13.639   0.635
    2p(2.00)    -5.414   0.635
    *s          13.573   0.635
    *p          21.797   0.635
    *d           0.000   0.635

  Using partial waves for C as LCAO basis

Reference energy: -244512.609073

Spin-paired calculation

Convergence criteria:
  Maximum total energy change: 0.001 eV / electron
  Maximum integral of absolute density change: 0.01 electrons
  Maximum integral of absolute eigenstate change: 0.0001 eV^2
  Maximum number of iterations: 16

Symmetries present (total): 4

  ( 1  0  0)  ( 1  0  0)  (-1  0  0)  (-1  0  0)
  ( 0  1  0)  ( 0  1  0)  ( 0 -1  0)  ( 0 -1  0)
  ( 0  0  1)  ( 0  0 -1)  ( 0  0  1)  ( 0  0 -1)
```

```
  Using the LDA Exchange-Correlation functional
  Interpolation: tri-quintic (5. degree polynomial)
  Poisson solver: FastPoissonSolver using
    6*3+1=19 point O(h^6) finite-difference Laplacian stencil;
    FFT axes: [2];
    FST axes: [0, 1].


Memory estimate:
  Process memory now: 94.34 MiB
  Calculator: 63.81 MiB
    Density: 3.43 MiB
      Arrays: 1.49 MiB
      Localized functions: 1.39 MiB
      Mixer: 0.54 MiB
    Hamiltonian: 1.09 MiB
      Arrays: 0.98 MiB
      XC: 0.00 MiB
      Poisson: 0.00 MiB
      vbar: 0.11 MiB
    Wavefunctions: 59.30 MiB
      Arrays psit_nG: 29.35 MiB
      Eigensolver: 29.63 MiB
      Projections: 0.13 MiB
      Projectors: 0.18 MiB

Total number of cores used: 100
Domain decomposition: 5 x 4 x 5

Number of atoms: 240
Number of atomic orbitals: 960
Number of bands in calculation: 540
Number of valence electrons: 960
Bands to converge: occupied

... initialized

Initializing position-dependent things.

Density initialized from atomic densities
Creating initial wave functions:
  540 bands from LCAO basis set
```



Example Benchmark

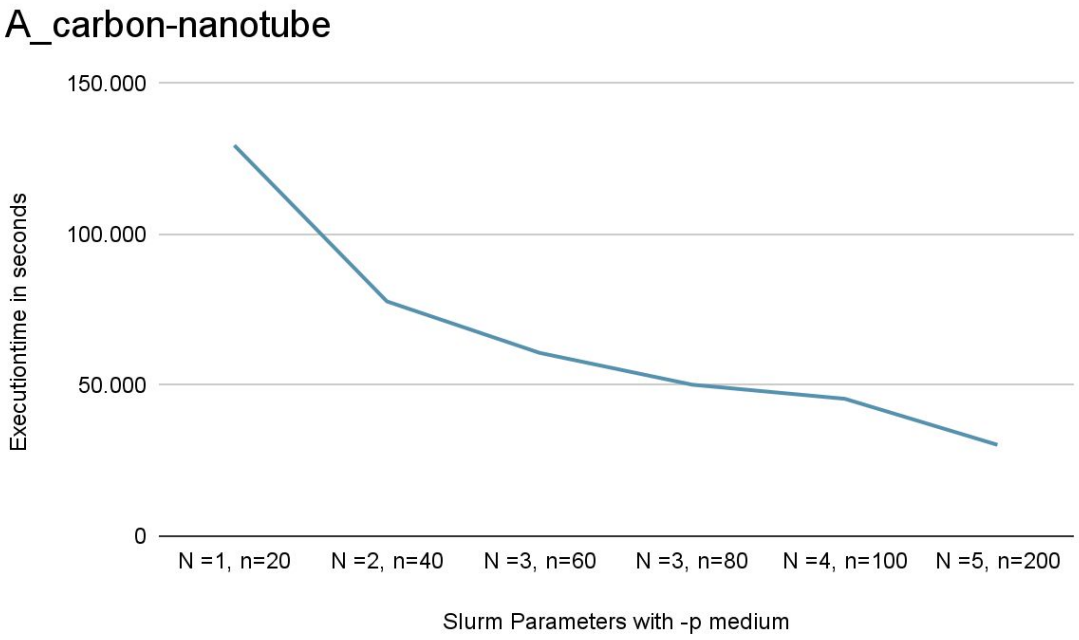
GPAW Example Output on dataset A_carbon-nanotube

```
#!/bin/bash
#SBATCH -p medium
#SBATCH -N 4
#SBATCH -n 100
#SBATCH -o job-%J.out

module purge
module load spack-user
source $SPACK_USER_ROOT/share/spack/setup-env.sh
spack load py-gpaw

srun gpaw python input.py
```

```
Result information:
  * Time:                    45.509 s
  * Number of iterations:    12
  * Dipole (3rd component): -115.165109
  * Fermi level:             -4.61200
  * Extrapolated energy:     -2397.625099
```

Example Benchmark

GPAW Example Output on dataset A_carbon-nanotube



References

- https://repository.prace-ri.eu/git/UEABS/ueabs
- https://hpc-wiki.info/hpc/Benchmarks
- https://en.wikipedia.org/wiki/Benchmark_(computing)#cite_note-5
- https://dberleant.github.io/papers/BenchmarkingContemporaryDeepLearningHardwareAndFrameworks.pdf